Survey: Anomaly Detection Methods

Ekta Gujral

Department of Computer Science and Engineering
University of California Riverside
egujr001@ucr.edu

Abstract. Anomaly detection is an important problem that has been fundamentally researched in various research and application domains. This survey aims three-fold; firstly, we present a structured and comprehensive overview of anomaly detection methods. Secondly, we review the applications and their effectiveness across various domains. Furthermore, we provide a practical implementation and use of these methods. We grouped current methods into different categories based on the fundamental approach. For each category, we discuss key assumptions used by the methods to differentiate between normal and abnormal behavior. The automatic detection and alerting of abnormal data and behaviors, implemented using computationally efficient software, are critical in this context. These considerations motivate applying the anomaly detection methods to real-world data discussed in this survey. This outline provides an easier and more brief understanding of the methods belonging to each category. We hope this survey will provide a great understanding of the various directions in which research has been done on this topic and how methods developed in one area can be applied in other domains.

1 Introduction

Anomaly detection [5] is a technique used in data analysis and machine learning to identify rare or unusual events, observations, or patterns that do not conform to the expected behavior or norm within a dataset. It involves finding patterns that deviate from the normal behavior of the data, which can indicate potential problems, outliers, or anomalies that require further investigation. Anomaly detection is a critical task in many fields, including cyber-security, finance, healthcare, and industrial process control. **Anomaly detection refers to the process of identifying patterns in data that deviate from the expected or normal behavior.** These patterns may indicate malicious activity, equipment failure, or other forms of abnormal behavior that can have significant consequences.

The detection of anomalies is a challenging problem because anomalies can take many different forms and can be difficult to distinguish from normal behavior. Anomalies can occur at any level of abstraction, from individual data points to entire systems, and can be caused by a wide range of factors, including human error, software bugs, hardware failures, and malicious activity. To address these challenges, a wide range of anomaly detection techniques have been developed, including statistical methods, machine learning algorithms, and deep learning models. These techniques can be broadly classified into supervised, unsupervised, and semi-supervised methods, depending on the availability of labeled data. Supervised methods require labeled data that is used to train a model to distinguish between normal and anomalous behavior. Unsupervised methods, on the other hand, do not require labeled data and instead use statistical techniques or clustering algorithms to identify patterns in the data that deviate from the expected behavior. Semi-supervised methods are a hybrid approach that combines elements of both supervised and unsupervised methods.

The choice of anomaly detection technique depends on a variety of factors, including the type of data being analyzed, the nature of the anomalies, and the available resources for training and deployment. In recent years, deep learning models, such as autoencoders and recurrent neural networks, have shown promise for detecting anomalies in complex data, such as images and time-series data.

The process of anomaly detection usually involves the following steps:

- **Data collection and pre-processing**: Collect the data from various sources, clean and pre-process it to make it ready for analysis.
- **Feature extraction**: Identify the relevant features or attributes that can be used to represent the data.
- **Model training**: Train a model using the labeled data or unsupervised learning techniques to identify normal behavior or patterns within the data.
- **Anomaly detection**: Apply the trained model to identify the observations or events that deviate from the normal behavior.
- **Evaluation and refinement**: Evaluate the results, refine the model if necessary, and repeat the process until the desired level of accuracy is achieved.

Overall, the detection of anomalies is a critical task for ensuring the security, reliability, and safety of systems in a variety of domains. Anomaly detection techniques continue to evolve and improve, driven by advances in statistical analysis, machine learning, and deep learning, and are expected to play an increasingly important role in the coming years.

1.1 Applications of Anomaly Detection

Anomaly detection has various applications in many fields such as fraud detection, intrusion detection, fault detection, network monitoring, and predictive maintenance. Here are some examples:

- **Cybersecurity**: Anomaly detection is widely used in cybersecurity to identify suspicious network activity, such as unusual logins, file transfers, or data access patterns. For example, intrusion detection systems (IDS) use anomaly

detection techniques to identify attacks or suspicious behavior on computer networks.
- **Manufacturing**: Anomaly detection can be used in manufacturing to monitor machines and equipment to detect any deviations from their normal behavior. This can help to identify potential faults or defects in the production process and avoid downtime. For example, vibration sensors can be used to monitor the health of machines and detect any unusual vibrations that may indicate a fault.
- **Finance Fraud detection**: Anomaly detection is used in finance to detect fraudulent transactions or unusual behavior in financial transactions. For example, credit card companies use anomaly detection to identify transactions that are outside the normal spending patterns of their customers.
- **Healthcare**: Anomaly detection is used in healthcare to identify unusual patterns in patient data that may indicate potential health issues or anomalies. For example, anomaly detection can be used to identify patients who are at high risk of developing a particular disease based on their medical history and other risk factors.
- **Energy and Utilities**: Anomaly detection can be used in energy and utilities to detect anomalies in power consumption, such as sudden spikes or drops in usage. This can help to identify potential faults in the power grid and take appropriate action to avoid blackouts or other disruptions.
- **Industrial process monitoring**: Anomaly detection can be used to monitor industrial processes for abnormal behavior that may indicate equipment failure or production issues.
- **Predictive maintenance**: Anomaly detection can be used to detect anomalies in machine sensor data to predict equipment failure and perform maintenance proactively.
- **Quality control**: Anomaly detection can be used to monitor manufacturing processes for defects or anomalies that may affect product quality.
- **Energy management**: Anomaly detection can be used to monitor energy consumption patterns in buildings and detect anomalies that may indicate energy waste or inefficiency.
- **Social media analysis**: Anomaly detection can be used to detect anomalous behavior in social networks, such as spamming or bot activity.
- **Traffic management**: Anomaly detection can be used to monitor traffic patterns and detect anomalies that may indicate accidents or congestion.
- **Environmental monitoring**: Anomaly detection can be used to detect anomalies in environmental data, such as air or water quality, to identify potential hazards or pollution sources.

These are just a few examples of how anomaly detection can be used in various domains. In general, anomaly detection can be used in any situation where there is a need to identify unusual patterns or behavior within a dataset.

1.2 Types of Anomalies

The type of intended anomaly is a key factor in an anomaly detection technique. The three categories of anomalies are as follows:

- **Point Anomaly**: An instance of data is referred to as a point anomaly if it may be considered abnormal concerning the rest of the data. A key characteristic of point anomaly (Fig. 2) is the return of the data to its previous normal state within a very short time period. These point anomalies may represent statistical noise, an irregularity, deviation that happens randomly or they could represent a significant short period event that is of interest to the business. Because of its simplicity, majority of the research is focused on point anomaly detection. For example, a single balmy day in an otherwise chilly winter would be a good example of this. On that day, the weather is considered anomalous because the temperature is extreme compared to the rest of the season. Point anomalies often occur in this way, as a singular extreme value on a single attribute of the data.
  Another business use case example, consider item sales anomaly detection. Let the data set correspond to each item's sales time series. For the sake of simplicity, let us assume that the data is defined using only one feature: total sales per day. A time stamp for which the sale is very high compared to the normal range of sales per day for that item will be a point anomaly.
- **Contextual Anomalies**: Contextual anomalies or conditional outliers are observations or sequences which deviate from the expected patterns within the data. However, if taken in separation they may be within the range of expected values. In simple words, a contextual anomaly (Fig. 1) is a deviation from the norm. This is referred to as contextual anomaly, where the data instance has to have some feature(s) that pertain to the context [2], whether it is time-relevant (i.e. temporal), location-relevant (i.e. spatial), or a different kind of context per the problem domain. For example, having a staff member attempt to log in to her corporate system using her credentials is not anomalous. However, when this does not happen within the pre-defined business hours, the instance becomes anomalous in the temporal context. Fig. 2 illustrates an example of a contextual anomaly. Another example, one of your customers may double their usual spending behavior in mid-December for the holiday season. These outliers are common in time series data because those datasets are records of specific quantities for given periods.
  - **Contextual attributes**: The contextual attributes are used to determine the context for that data instance. For example, in spatial data sets, the longitude and latitude of a location are the contextual attributes. In time-series data, time is a contextual attribute which determines the position of an instance on the entire sequence. An object in a given data set is a contextual outlier (or conditional outlier) if it deviates significantly with respect to a specific context of the object.
  - **Behavioral attributes**: The behavioral attributes determine the non-contextual characteristics of an instance or object and are used to evalu-

ate whether the instance or object is an outlier in the context to which it belongs. For example, in a spatial data set describing the average rainfall of the entire world, the amount of rainfall at any location is a behavioral attribute.

– **Collective Anomalies**. A collective anomaly is a collection of similar data points that can be considered abnormal together when compared to the rest of the data. For example, a consecutive 10-day period of hot temperatures could be considered a collective anomaly. These temperatures are unusual because they occur together and are likely caused by the same underlying weather event.

1.3 Challenges

The development of an anomaly detector goes through several phases. The process begins with understanding data, model creation, validation, and testing, followed by deployment and tuning, and lastly operation and retraining. But several factors make this apparently simple process very challenging:

Data quality : When developing an anomaly detection model, one primary question we may ask our self is: "Which algorithm should I use for my application?" This is highly dependent on the type of problem or application we're trying to solve, of course, but main thing to think about is the quality of underlying data and availability of labeled data. Often the data contains noise which tends to be similar to the actual anomalies and hence is difficult to distinguish and remove. Data quality problems can include but not limited to null data or incomplete datasets, inconsistent data formats, duplicate data, different scales of measurement, and human error. Low data quality and the existence of noise carry a huge challenge to anomaly detection. They can deceive the information, blurring the differentiation among normal objects and anomaly. Furthermore, noise and missing information can "hide" outliers and decrease the effectiveness of anomaly detection an anomaly can occur "disguised" as a noise point, and an anomaly detection approach can erroneously recognize a noise point as an anomaly.

Imbalanced distributions : Another method of building an anomaly detection model would be to use a classification algorithm to build a supervised model. This supervised model will require labeled data to understand what is good or bad. A common problem with labeled data is distribution imbalance. It's normal to have a good state which means 99% of the labeled data will be skewed towards good. Because of this natural imbalance, the training set may not have enough examples to learn and associate with the bad state. This is another problem that can be hard to solve.

Modeling normal objects and outliers effectively : Anomaly detection element largely based on the modeling of normal objects and outliers. This is

slightly because it is complex to enumerate some available normal behaviors in an application. The border among data normality and abnormality is not clear cut. Instead, there can be a broad range of gray application. Consequently, while various anomaly detection techniques assign to each object in the input information set a label of either "normal" or "abnormal," other approach assign to each object a score calculating the "anomaly" of the object. Defining a normal region which encompasses every possible normal behavior is very difficult. In addition, the boundary between normal and anomalous behavior is often not precise. Thus an anomalous observation which lies close to the boundary can actually be normal, and vice-versa. When anomalies are the result of malicious actions, the malicious adversaries often adapt themselves to make the anomalous observations appear like normal, thereby making the task of defining normal behavior more difficult.

Domain-specific anomaly detection : In many domains normal behavior keeps evolving and a current notion of normal behavior might not be sufficiently representative in the future. The exact notion of an anomaly is different for different application domains. For example, in the medical domain a small deviation from normal (e.g., fluctuations in body temperature) might be an anomaly, while similar deviation in the stock market domain (e.g., fluctuations in the value of a stock) might be considered as normal. Thus applying a technique developed in one domain to another is not straightforward.

Handling noise : Anomalies are different from noise. It is known that the quality of real information sets influence to be poor. Noise provide unavoidably exists in data collected in several applications. Noise can be show as deviations in attribute values or make smooth as missing values.

Intelligible : In some application methods, a user can required to not only detect anomalies, but also learn why the detected objects are anomalies. It can combine the intelligible requirement, an anomaly detection techniques has to support some reasons of the detection. For instance, a statistical approach can be used to validate the degree to which an object can be a anomaly depends on the likelihood that the object was created by the same structure that generated the majority of the records. The smaller the likelihood, the more unlikely the object was produced by the same structure, and the more acceptable the object is a anomaly.

High dimensionality : Many datasets are high-dimensional, making it difficult to detect anomalies effectively as traditional statistical techniques may not scale well to higher dimensions.

Training sample sizes : Having a large training set is important for many reasons. If the training set is too small, then the algorithm does not have enough

exposure to past examples to build an accurate representation of the expected value at a given time. Anomalies will skew the baseline, which will affect the overall accuracy of the model. Seasonality is another common problem with small sample sets. Not every day or week is the same, which is why having a large enough sample dataset is important. Customer traffic volumes may spike during the holiday season, or could significantly drop depending on the line of business. It's important for the model to see data samples for multiple years so it can accurately build and monitor the baseline during common holidays.

False alerting : Identifying anomalies is an excellent tool in a dynamic environment as it can learn from the past to identify expected behavior and anomalous events. But what happens when your model continuously generates false alerts and is consistently wrong? It's hard to gain trust from skeptical users and easy to lose it, which is why it's important to ensure a balance in sensitivity.

No clear definition of what constitutes an anomaly : Anomalies can be subjective, and what one considers to be an anomaly may differ from another. This can lead to difficulties in defining and identifying anomalies accurately.

Lack of labeled data : Anomaly detection typically requires labeled data to train models. However, in many real-world scenarios, anomalous data may be rare or difficult to obtain, making it challenging to train effective models.

Concept drift : Anomalies can change over time, which can make it difficult to detect them accurately. As such, anomaly detection models may need to be updated regularly to adapt to changing conditions.

Scalability : As datasets become larger and more complex, it can be challenging to develop anomaly detection methods that can scale effectively to these larger datasets.

Interpretability : Many anomaly detection models are complex and difficult to interpret, making it difficult to understand how they arrive at their conclusions. This can make it challenging to determine the causes of anomalous behavior and develop appropriate responses.

Due to the above challenges, the anomaly detection problem, in its most general form, is not easy to solve. In fact, most of the existing anomaly detection techniques solve a specific formulation of the problem. The formulation is induced by various factors such as nature of the data, availability of labeled data, type of anomalies to be detected, etc. Often, these factors are determined by the application domain.

2 What are Anomaly Detection methods?

We provide a details of various anomaly detection techniques together with short descriptions of the different algorithm types and provides general ideas behind the algorithms. More specifically, anomaly detection algorithms [17,24] are based on Classic Machine Learning, Deep Learning, Stochastic Learning, Outlier Detection, Statistics, Matrix Profiling, Data Mining, and Signal Analysis.Implemenations of below methods are available at link[1]

2.1 Classic Machine Learning Based Anomaly Detection

K-Means [9,21,12,13,25]: The k-Means which is firstly proposed by James MacQueen [9], is a well-known and widely used clustering algorithm and do not require training data for anomaly detection. It is one of the simplest clustering algorithms in machine learning which can be used to automatically recognize groups of similar instances in data. In this method, K random points are selected as centroids in a dataset. Then, the elements are arranged to the closest centroids by calculating the distance. The process is repeated to achieve optimal distances between sample data and centroids. We want the inter-cluster distance to be large, while the intra-cluster distance to be small. By achieving this, the groups are more distinguishable, and the subjects within a single group are more alike.

Hybrid K-Means [23]: K-Means algorithm is one of the most efficient partitioning clustering algorithm. The algorithm is simple and easy to implement, suitable for large data sets, and very highly active. However, it has two major drawbacks: 1) the number of randomly chosen points and the centroid of clusters may lead to different clustering results, 2) K-Means algorithm may contain many local convergence. In some of the previous anomaly detection systems have adopted the K-Means clustering, but the result is not ideal. There are a lot of clusters and local optima, this algorithm run several times will get different results. In order to overcome these shortcomings of K-Means, [23] use the Particle Swarm Optimization (PSO) to combine the K-Means algorithm. PSO is a heuristic algorithm, it can be a minimum number of iterations to find the optimal or near-optimal solutions. PSO can find good initial cluster centers by its' global search capability and then avoid K-Means algorithm falling into local optimal solution.

KNN [16]: k-NN is one of the simplest methods in machine learning. It classifies a data point based on how its neighbors are classified. So if we are guessing the color of a particular wine, we can refer to the color of other wines that have the most similar chemical makeup (i.e. neighbors). Besides classification into groups, k-NN can also be used to estimate continuous values. To approximate a data point's value, k -NN takes the aggregated value of its most similar neighbors. k-NN is not limited to merely predicting groups or values of data points. It can

also be used in detecting anomalies. Identifying anomalies can be the end goal in itself, such as in fraud detection. Anomalies can also lead you to additional insights, such as discovering a predictor you previously overlooked. Although k-NN is simple and effective, there are times when it might not work as well. If there are multiple classes to be predicted, and the classes differ drastically in size, data points belonging to the smallest class might be overshadowed by those from bigger classes, increasing their risk of mis-classification. To improve accuracy, we could use swap majority voting in favor of weighted voting, whereby the class of closer neighbors are weighted more heavily than ones further away. If there are too many predictors to consider, it would be computationally intensive to identify and process nearest neighbors across multiple dimensions. Moreover, some predictors could be redundant as they do not improve prediction accuracy. To resolve this, we can use dimension reduction techniques to extract only the most powerful predictors for analysis.

PCA [19]: Principal Component Analysis (PCA) is a popular technique for anomaly detection in multivariate datasets. The basic idea behind PCA-based anomaly detection is to reduce the dimensionality of the data while retaining as much of the variance as possible. By projecting the data onto a lower-dimensional space, anomalies can be identified as data points that are distant from the majority of the data. The basic steps for performing anomaly detection using PCA:

- Normalize the data: Standardize the data by subtracting the mean and dividing by the standard deviation to ensure that all variables are on the same scale.
- Compute the principal components: Use PCA to compute the principal components of the data, which are the directions in which the data has the most variance.
- Compute the reconstruction error: Project the data onto the principal components and then reconstruct the original data. The reconstruction error is the difference between the original data and the reconstructed data.
- Compute the threshold: Determine a threshold value for the reconstruction error above which a data point is considered an anomaly. The threshold can be set using statistical techniques such as the mean or median, or by using domain-specific knowledge.
- Identify anomalies: Identify data points whose reconstruction error exceeds the threshold as anomalies.

In [19], the paper presents a novel anomaly detection scheme based on a principal component classifier (PCC). The proposed scheme uses the PCC to project data onto a low-dimensional space, where anomalies can be easily detected using statistical methods. The PCC is trained on a set of normal data points and is then used to classify new data points as either normal or anomalous. The scheme is evaluated using several benchmark datasets, and the results show that it outperforms existing state-of-the-art anomaly detection methods in terms of both detection accuracy and computational efficiency. PCA-based anomaly detection can be used for many applications such as fraud detection, intrusion

detection, and fault detection. However, it has limitations, such as being sensitive to outliers and assuming that the data follows a normal distribution. Other methods such as robust PCA or kernel PCA can be used to overcome these limitations.

RobustPCA [15] Robust Principal Component Analysis (RPCA) is a method used for anomaly detection in data. It is based on the assumption that the data can be decomposed into a low-rank matrix and a sparse matrix. The low-rank matrix contains the underlying structure or regularities in the data, while the sparse matrix contains the anomalous or outlier data points. The main idea behind RPCA is to solve the following optimization problem:

$$\text{minimize } ||L||_* + \lambda||S||_1 \text{subject to } A = L + S \tag{1}$$

where A is the data matrix, L is the low-rank matrix, S is the sparse matrix, λ is a regularization parameter, $||.||_*$ denotes the nuclear norm (i.e., the sum of the singular values), and $||.||_1$ denotes the L1 norm. The solution to this optimization problem can be obtained using various algorithms, such as alternating direction method of multipliers (ADMM), accelerated proximal gradient (APG), and robust principal component pursuit (RPCP). Once the low-rank and sparse matrices are obtained, the anomalous data points can be identified as those with large values in the sparse matrix. In [26], the paper proposes a robust principal component analysis (RPCA) approach for anomaly detection in cyber networks. The proposed method first performs robust PCA to extract the low-dimensional structure of the data and then applies a clustering algorithm to identify anomalies. The method is evaluated using several datasets, and the results show that it outperforms existing state-of-the-art anomaly detection methods, particularly in detecting outliers in high-dimensional data.

RPCA has been successfully applied in various applications [11,10,7,26,18], such as image and video processing, sensor networks, and cybersecurity. However, it may not work well in all cases, especially when the data has complex structures or the anomalies are not sparse. In such cases, other anomaly detection methods may be more appropriate. RPCA approach has been shown to be effective in detecting credit card fraud and has been used by many financial institutions and credit card companies. However, it is important to note that the performance of the algorithm depends on the quality of the data and the choice of parameters such as the regularization parameter λ. Therefore, it is recommended to tune the parameters carefully and evaluate the algorithm on a validation set to ensure its effectiveness in detecting anomalies.

2.2 Isolation forest

[14]: Isolation Forest is an algorithm for detecting anomalies or outliers in a dataset. The basic idea behind Isolation Forest is to build many decision trees, with each tree splitting the data into smaller and smaller subsets. The algorithm isolates anomalies by considering how many splits are required to isolate each

data point. Anomalies are considered to be data points that require fewer splits to be isolated than normal data points. Isolation Forest has several advantages over other outlier detection algorithms. It is highly scalable, as it can handle large datasets with many features. It is also relatively simple to implement, as it does not require any complicated algorithms or statistical models. Additionally, Isolation Forest does not make any assumptions about the distribution of the data, making it suitable for a wide range of datasets. However, like any algorithm, Isolation Forest has its limitations. It may not perform well in cases where the data has a complex distribution or where the anomalies are not easily isolated. In such cases, other algorithms, such as support vector machines or neural networks, may be more appropriate.

Random Forest Regressor [20]: Anomaly detection using Random Forest Regressor involves training a regression model on a dataset and using it to predict the expected output for each data point. If a data point has a significantly different output than what is predicted by the model, it can be considered an anomaly or outlier. Suppose we have a dataset containing the performance metrics of a set of servers. The dataset contains various features such as CPU utilization, memory usage, disk I/O, and network traffic. We want to identify servers that are performing abnormally or are likely to fail soon. First, we split the dataset into a training set and a test set. The training set is used to train the Random Forest Regressor model, while the test set is used to evaluate its performance. Next, we train the Random Forest Regressor model on the training set. The model is trained to predict the expected output (performance metrics) for each server based on its input features. Once the model is trained, we use it to predict the output for each data point in the test set. We compare the predicted output with the actual output for each data point. If the difference between the predicted and actual output is higher than a certain threshold, we flag the data point as an anomaly. Finally, we analyze the flagged data points to determine if they are actually anomalous or if there was some issue with the data. Random Forest Regressor is a popular algorithm for anomaly detection because it can handle high-dimensional datasets with complex relationships between the features. It also provides a measure of feature importance, which can be used to identify the most important features for detecting anomalies. However, it is important to note that Random Forest Regressor is not guaranteed to find all anomalies in the dataset and may require careful tuning of hyperparameters such as the number of trees and the maximum depth of the trees.

Random Black Forest Random Black Forest (RBF) is a variant of Random Forest algorithm that is specifically designed for anomaly detection. The RBF algorithm is based on the concept of random projection and clustering, which allows it to effectively detect anomalies in high-dimensional datasets. The RBF projects the high-dimensional input data onto a low-dimensional space using random projection and then cluster the data points based on their proximity in the low-dimensional space. The RBF algorithm is then used to detect anomalies

by analyzing the distribution of the cluster assignments for the data points. The RBF algorithm has been shown to be effective for anomaly detection in various applications, including cybersecurity, finance, and manufacturing. The algorithm is computationally efficient and can handle high-dimensional datasets with a large number of features. Moreover, the RBF algorithm does not require any prior knowledge of the data distribution or the types of anomalies that might exist in the data. Here are some papers that discuss the RBF algorithm and its applications:

- Breunig, M. M., Kriegel, H. P., Ng, R. T., and Sander, J. (2000). LOF: identifying density-based local outliers. In Proceedings of the 2000 ACM SIGMOD international conference on Management of data (pp. 93-104). ACM.[4]
- Liu, F. T., Ting, K. M., and Zhou, Z. H. (2008). Isolation forest. In 2008 Eighth IEEE International Conference on Data Mining (pp. 413-422). IEEE.[14]

2.3 Forecasting Based Anomaly Detection

Forecasting methods are a class of anomaly detection techniques that use time-series data to predict future values and identify anomalies based on deviations from the predicted values. Some common forecasting methods for anomaly detection include:

ARIMA (Autoregressive Integrated Moving Average) [3]: ARIMA, which stands for AutoRegressive Integrated Moving Average, is a time series forecasting method used for univariate time series data. It combines three components:

- Autoregression (AR): A model that uses the dependent relationship between an observation and a number of lagged observations.
- Integration (I): The use of differences of raw observations to make the time series stationary, i.e., to remove trends and seasonal effects.
- Moving Average (MA): A model that uses the dependence between an observation and a residual error from a moving average model applied to lagged observations.

The ARIMA model is fit to the time series data by finding the best parameters for the AR, I, and MA components, such that the difference between the observed values and the values predicted by the model is minimized. ARIMA models can be used for a variety of tasks, including out-of-sample forecasting, trend analysis, and anomaly detection. In practice, the process of selecting the appropriate ARIMA model involves determining the values of the parameters that best capture the underlying structure of the time series data. This can be a complex and time-consuming task, but there are automated methods, such as the "auto.arima" function in the R programming language, that can simplify the process.

Exponential Smoothing [8]: Exponential smoothing is a time series forecasting method that is used to forecast data points by employing a weighted average

of past observations. The method assigns exponentially decreasing weights as the observations get older, hence the name "exponential smoothing."

In exponential smoothing, the forecast for time t+1 is a weighted average of the observed values up to time t, where the weights decrease exponentially as the observations get older. The forecast is updated after each new observation is received, with the latest observation receiving the highest weight and older observations receiving progressively lower weights. There are several variants of exponential smoothing, including simple exponential smoothing, Holt's linear exponential smoothing, and Holt-Winters exponential smoothing. Each of these variants uses a different combination of trends and seasonality to produce a forecast. The choice of method depends on the characteristics of the time series data, such as whether it has a trend, seasonality, or both.

Exponential smoothing is a simple and fast method for time series forecasting and is often used for short-term forecasting. It is also relatively easy to implement and can be used for a wide range of applications, from financial forecasting to demand forecasting in supply chain management and anomaly detection. For example, if we have a time series of web traffic data, we can use exponential smoothing to model the typical patterns of web traffic over time. If a new data point deviates significantly from the model's predictions, it could indicate an anomaly, such as a sudden spike in traffic or a sudden drop. In practice, exponential smoothing can be combined with statistical tests, such as the Z-score or the chi-squared test, to determine the significance of deviations from the model. If a deviation is significant, it can be flagged as an anomaly.

Prophet [22]: Prophet is a popular open-source software library developed by Facebook for forecasting time series data. It is designed to handle many of the complexities of time series data, such as trends, seasonality, and holidays, and can be used for a wide range of applications, including sales forecasting, demand forecasting, and resource planning. Prophet is based on a Bayesian structural time series model and uses a decomposable time series model that separates the trend, seasonality, and holiday components of the data. The model then uses a flexible non-linear regression model to fit the data, taking into account the impact of holidays and seasonality on the time series.

One of the key features of Prophet is its ability to handle missing data, outliers, and large changes in the time series data. It also includes a built-in capability for modeling changepoints, which allows it to handle changes in the underlying trends of the data. Prophet is implemented in Python and can be easily integrated with other data analysis tools, such as Jupyter Notebooks, Pandas, and scikit-learn. It is also highly customizable, allowing users to specify custom holidays, changepoints, and other parameters to fit the data. Overall, Prophet is a flexible and user-friendly tool for time series forecasting that is well-suited for a wide range of applications. Its ability to handle many of the complexities of time series data and its ease of use make it a popular choice among data scientists and analysts.

LSTM (Long Short-Term Memory) [6]: LSTM is a type of recurrent neural network (RNN) that can model long-term dependencies in time-series data.

Seasonal Hybrid ESD (Extreme Studentized Deviation) [1]: Seasonal Hybrid ESD is a time-series anomaly detection algorithm that uses a combination of seasonal decomposition and statistical tests to identify anomalies in time-series data.

2.4 Graph Based

Graph-based anomaly detection is an approach that involves representing data as a graph, where nodes represent data points and edges represent relationships between the data points. In recent years, there has been growing interest in applying graph-based anomaly detection to a wide range of applications, including social network analysis, cybersecurity, and anomaly detection in time-series data.

One example of a graph-based anomaly detection algorithm is the Local Outlier Factor (LOF) algorithm. LOF is an unsupervised algorithm that measures the local density of a data point relative to its neighbors. Points that have a significantly lower density than their neighbors are considered anomalies. LOF has been applied to a variety of applications, including intrusion detection in network traffic data and anomaly detection in time-series data.

Another example of a graph-based anomaly detection algorithm is the Isolation Forest algorithm. Isolation Forest is a tree-based algorithm that isolates anomalies by randomly partitioning the data points and creating a set of trees. Anomalies are identified as data points that require a small number of partitions to isolate. Isolation Forest has been applied to a variety of applications, including fraud detection and intrusion detection in network traffic data.

Graph-based anomaly detection has also been applied to social network analysis, where anomalies can be identified based on changes in network structure or behavior. For example, the Graph Convolutional Network (GCN) algorithm can be used to identify anomalies in social network data by detecting changes in the connectivity patterns of nodes.

Overall, graph-based anomaly detection is a promising approach for identifying anomalies in a wide range of applications. Graph-based algorithms can capture complex relationships between data points and can often be applied to data with a high degree of variability and noise. As data continues to grow in complexity and volume, graph-based anomaly detection is expected to play an increasingly important role in detecting and preventing anomalous behavior.

3 Conclusion

The future of anomaly detection methods looks promising, as there is a growing need for effective and efficient ways to detect anomalous behavior in complex datasets. As data continues to grow in complexity and volume, traditional

anomaly detection methods may become less effective, and new approaches will need to be developed to meet the demands of emerging applications.

One area of research that shows promise is the use of deep learning and artificial intelligence techniques for anomaly detection. These methods have shown impressive results in detecting anomalies in image, speech, and natural language data, and they are expected to be increasingly applied to other domains.

Another promising area of research is the use of unsupervised learning techniques for anomaly detection. Unsupervised learning methods can detect anomalies without the need for labeled data, making them applicable to a wider range of applications. One example of an unsupervised learning method is Generative Adversarial Networks (GANs), which can generate new samples of data and detect anomalies based on the differences between the generated and real data.

Graph-based anomaly detection methods also show promise, as they can capture complex relationships between data points and can be applied to a wide range of applications, including social network analysis and cybersecurity.

In addition, the integration of anomaly detection methods with other machine learning techniques, such as clustering and classification, is expected to lead to more robust and accurate anomaly detection systems.

Overall, the future of anomaly detection methods looks bright, as researchers and practitioners continue to develop new and innovative approaches to meet the challenges of detecting anomalous behavior in complex datasets. With the increasing importance of data-driven decision making in a wide range of industries, the need for effective anomaly detection methods will only continue to grow in the years to come.

References

1. G. Amati, S. Angelini, G. Gambosi, D. Pasquin, G. Rossi, and P. Vocca. Twitter: temporal events analysis. In *Proceedings of the 4th EAI International Conference on Smart Objects and Technologies for Social Good*, pages 298–303, 2018.
2. S. W. A.-H. Baddar, A. Merlo, and M. Migliardi. Anomaly detection in computer networks: A state-of-the-art review. *J. Wirel. Mob. Networks Ubiquitous Comput. Dependable Appl.*, 5(4):29–64, 2014.
3. G. E. Box and D. A. Pierce. Distribution of residual autocorrelations in autoregressive-integrated moving average time series models. *Journal of the American statistical Association*, 65(332):1509–1526, 1970.
4. M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander. Lof: identifying density-based local outliers. In *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, pages 93–104, 2000.
5. V. Chandola, A. Banerjee, and V. Kumar. Anomaly detection: A survey. *ACM computing surveys (CSUR)*, 41(3):1–58, 2009.
6. S. Didari, T. Liao, and H. Rajagopal. Long short-term memory anomaly detection for multi-sensor equipment monitoring, Apr. 2 2020. US Patent App. 16/580,761.
7. H. Ding, W. Fan, and X. Wang. Anomaly detection in surveillance video using pca and svm. *Journal of Signal Processing Systems*, 81(3):357–370, 2015.
8. E. S. Gardner Jr. Exponential smoothing: The state of the art—part ii. *International journal of forecasting*, 22(4):637–666, 2006.

9. M. James. Some methods for classification and analysis of multivariate observations.[in] proceedings of 5-th berkeley symposium on mathematical statistics and probability. 1967.
10. K. Kamal, I. Ahmad, A. Razaque, and M. Adnan. Anomaly detection using pca in network traffic. *International Journal of Computer Applications*, 180(45):13–17, 2019.
11. S. Kim, W. Kim, K. Park, and G. Jang. Anomaly detection using principal component analysis for healthcare system. *Sensors*, 18(4):1164, 2018.
12. R. Kumari, Sheetanshu, M. K. Singh, R. Jha, and N. Singh. Anomaly detection in network traffic using k-mean clustering. In *2016 3rd International Conference on Recent Advances in Information Technology (RAIT)*, pages 387–393, 2016.
13. M. F. Lima, B. B. Zarpelao, L. D. Sampaio, J. J. Rodrigues, T. Abrao, and M. L. Proença. Anomaly detection using baseline and k-means clustering. In *SoftCOM 2010, 18th International Conference on Software, Telecommunications and Computer Networks*, pages 305–309. IEEE, 2010.
14. F. T. Liu, K. M. Ting, and Z.-H. Zhou. Isolation forest. In *2008 eighth ieee international conference on data mining*, pages 413–422. IEEE, 2008.
15. R. Paffenroth, K. Kay, and L. Servi. Robust pca for anomaly detection in cyber networks. *arXiv preprint arXiv:1801.01571*, 2018.
16. S. Ramaswamy, R. Rastogi, and K. Shim. Efficient algorithms for mining outliers from large data sets. In *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, pages 427–438, 2000.
17. S. Schmidl, P. Wenig, and T. Papenbrock. Anomaly detection in time series: A comprehensive evaluation. 15(9):1779–1797.
18. H. Shi, L. Fan, Z. Liu, and W. Qiu. Pca-based anomaly detection for smart grid security. *IEEE Access*, 7:29577–29589, 2019.
19. M.-L. Shyu, S.-C. Chen, K. Sarinnapakorn, and L. Chang. A novel anomaly detection scheme based on principal component classifier. Technical report, Miami Univ Coral Gables Fl Dept of Electrical and Computer Engineering, 2003.
20. H. Smaili, J. Breeman, T. Lombaerts, and D. Joosten. A simulation benchmark for integrated fault tolerant flight control evaluation. In *AIAA modeling and simulation technologies conference and exhibit*, page 6471, 2006.
21. I. Syarif, A. Prugel-Bennett, and G. Wills. Unsupervised clustering approach for network anomaly detection. In *Networked Digital Technologies: 4th International Conference, NDT 2012, Dubai, UAE, April 24-26, 2012. Proceedings, Part I 4*, pages 135–145. Springer, 2012.
22. S. J. Taylor and B. Letham. Forecasting at scale. *The American Statistician*, 72(1):37–45, 2018.
23. K.-W. Wang and S.-J. Qin. A hybrid approach for anomaly detection using k-means and pso. In *2nd International Conference on Electronics, Network and Computer Engineering (ICENCE 2016)*, pages 821–826. Atlantis Press, 2016.
24. P. Wenig, S. Schmidl, and T. Papenbrock. Timeeval: A benchmarking toolkit for time series anomaly detection algorithms. 15(12):3678–3681.
25. T. Yairi, Y. Kato, and K. Hori. Fault detection by mining association rules from house-keeping data. In *proceedings of the 6th International Symposium on Artificial Intelligence, Robotics and Automation in Space*, volume 18, page 21. Citeseer, 2001.
26. C. Zhou, X. Lu, and X. Liu. Robust pca for anomaly detection in cyber networks. In *2017 IEEE Conference on Communications and Network Security (CNS)*, pages 1–9. IEEE, 2017.